Modellbasierte Spezifikation von RESTful SOA auf Basis von Geschäftsprozessmodellen

Matthias Wolf

Veröffentlicht in:
Multikonferenz Wirtschaftsinformatik 2012
Tagungsband der MKWI 2012
Hrsg.: Dirk Christian Mattfeld; Susanne Robra-Bissantz

Braunschweig: Institut für Wirtschaftsinformatik, 2012

Modellbasierte Spezifikation von RESTful SOA auf Basis von Geschäftsprozessmodellen

Matthias Wolf

Universität Bamberg, Lehrstuhl für Wirtschaftsinformatik,
insbesondere Systementwicklung und Datenbankanwendung,
96045 Bamberg, E-Mail: matthias.wolf@uni-bamberg.de

Abstract

Der Einsatz von Web-Services auf Basis des Architekturstils REpresentational State Transfer (REST) gewinnt zur Realisierung service-orientierter Architekturen (SOA) immer mehr an Bedeutung. Im unternehmerischen Umfeld dienen SOAs häufig der Automatisierung flexibler Geschäftsprozesse. Mit dem zunehmenden Einsatz von RESTful Web-Services zur Geschäftsprozessautomatisierung wächst auch der Bedarf nach einem Ansatz zur Spezifikation einer RESTful SOA aus fachlichen Anforderungen. Diese werden häufig mit Geschäftsprozessmodellen analysiert, gestaltet und dokumentiert. Der vorliegende Beitrag erarbeitet einen dreistufigen Entwicklungsprozess für die modellbasierte Ableitung von RESTful SOA-Spezifikationen aus Geschäftsprozessmodellen.

1 Einleitung und Motivation

Unternehmen sehen sich in einer dynamischen Wirtschaft zunehmend hohen Anforderungen bezüglich Flexibilität und Wirtschaftlichkeit ausgesetzt. Die Bewältigung dieser Anforderungen erfolgt unter anderem durch die Gestaltung flexibler Geschäftsprozesse sowie der Entwicklung moderner betrieblicher Anwendungssysteme (AwS) (vgl. [7]). Moderne AwS werden in der Regel als verteilte Systeme nach dem Konzept der service-orientierten Architektur (SOA) realisiert. Mit dem SOA Konzept wird die Entwicklung lose-gekoppelter IT-Systeme angestrebt, deren Funktionalität als Dienste (Services) bereitgestellt, flexibel organisiert und verwendet werden können (vgl. [10]). Dieser Ansatz ist zunächst unabhängig vom Einsatz einer bestimmten Technologie. Jedoch erfolgt eine Realisierung gewöhnlich auf Basis von Web-Service-Technologien wie SOAP, WSDL und UDDI (vgl. [5], [11], [12]). Als interessante Alternative hierzu rücken in den letzten Jahren Web-Services auf Basis des Architekturstils REpresentational State Transfer (REST) immer weiter in den Fokus (vgl. [12], [18], [20]). Zum Architekturstil REST konforme Web-Services werden als RESTful Web-Services (REST-WS) bezeichnet (vgl. [12], [14]). Service-orientierte Architekturen, die auf Basis von REST-WS realisiert sind, werden im weiteren Verlauf als RESTful SOA bezeichnet. Ausführliche Vergleiche von REST-WS und SOAP/WSDL-Services finden sich in [12], [14] und [18].

Dort werden die Vorteile des Einsatzes von REST-WS im betrieblichen Umfeld hinsichtlich Flexibilität, Interoperabilität und Skalierbarkeit für adhoc Web-Integrationsszenarien gesehen.

Das Fehlen eines ganzheitlichen methodischen Ansatzes zur Spezifikation von RESTful SOA in betrieblichen Systemen wird in der Literatur als Problem genannt (vgl. [9], [12]). Insbesondere eine methodisch gestützte Identifikation von Ressourcen als zentrale REST-Systemkomponente auf Basis der in einem Unternehmen durchzuführenden Aufgaben wurde bisher wenig untersucht (zum Begriff Ressource siehe Abschnitt 2.1). Vorschläge, wie die Identifikation von REST-Ressourcen erfolgen kann, liegen meist nur natürlich-sprachig oder in Form von Best Practices vor (vgl. [1], [14], [18]). Zudem konzentrieren sich bestehende Ansätze auf die Implementierung von REST-WS sowie deren softwaretechnische Beschreibung, und weniger auf die Ableitung RESTful SOA aus einem fachlichen Entwurf (vgl. [1], [4], [14], [18]).

Im vorliegenden Beitrag wird ein modellbasierter Ansatz konzipiert, um die methodische Lücke zwischen fachlicher Analyse und softwaretechnischer Spezifikation von RESTful SOA in betrieblichen Systemen zu verringern. Im Fokus der Untersuchung steht dabei, wie eine methodisch gestützte Ableitung von REST-Ressourcen sowie eine durchgängig modellbasierte softwaretechnische Spezifikation einer RESTful SOA aus Aufgabenspezifikationen erfolgen kann. Ausgangspunkt des Ansatzes bilden dabei die informationsverarbeitenden Aufgaben, die in einem Unternehmen durchzuführen sind. Diese werden häufig mit Geschäftsprozessmodellen analysiert, gestaltet und dokumentiert. In der Praxis stützt sich die Entwicklung von REST-WS auf objektorientierte Entwurfsmethoden und Konzepte zur Entwicklung verteilter Systeme (vgl. [14], [18]). Diese Konzepte finden in der SOM-Methodik Berücksichtigung, die die durchgängige Entwicklung verteilter objektorientierter Systeme ausgehend von Geschäftsprozessmodellen beschreibt (vgl. Kapitel 3). Aus diesem Grund wird die SOM-Methodik für die Entwicklung der RESTful SOA gewählt.

In Kapitel 2 werden zunächst für die Arbeit grundlegende Begriffe zu REST und ein Vorgehen für die Entwicklung von REST-WS erläutert. Anschließend wird der konzipierte Entwicklungsprozess vorgestellt (Kapitel 3) und zur Veranschaulichung des Ansatzes auf ein Fallbeispiel angewendet (Kapitel 4). Der Beitrag schließt mit einem Verweis auf vergleichbare Ansätze (Kapitel 5), einer zusammenfassenden Diskussion der Ergebnisse und dem Ausblick auf weitere Forschungsaktivitäten (Kapitel 6).

2 Grundlagen

In diesem Kapitel werden für den weiteren Verlauf der Arbeit relevante Grundlagen beleuchtet. Vorgestellt werden wichtige REST-Konzepte sowie ein etabliertes Vorgehen für die Entwicklung von REST-WS.

2.1 RESTful Web-Services

Roy Fielding stellt in seiner Dissertation REST als Architekturstil für verteilte Systeme vor. Es wird spezifiziert wie existierende Web-Protokolle verwendet werden können, um u.a. Web-Services zu realisieren (vgl. [8]). Der Architekturstil REST wird anhand von vier Prinzipien charakterisiert (vgl. [8], [14], [18]): (1) Konzeptuelle Objekttypen und ihre Funktionalität werden als Ressourcen mit eindeutiger Identifikation (URL) entworfen. (2) Jede Ressource stellt eine standardisierte Schnittstelle mit den http-Methoden (get, post, put und delete) bereit. (3) Die

Interaktion mit der REST-Anwendung, beispielsweise durch andere Web-Services oder menschliche Nutzer, erfolgt über die Kommunikation und Manipulation der Repräsentation von Ressourcen. Eine Ressource kann mehrere Repräsentationen aufweisen. Eine Ressource Auftrag kann beispielsweise einem menschlichen Nutzer als HTML-Seite und einem AwS als maschinenlesbares XML-Dokument präsentiert werden. (4) Beziehungen zwischen und Interaktionen mit Ressourcen werden über Verknüpfungen (Links) realisiert. Zusammen mit den Prinzipien der eindeutigen Identifizierbarkeit von Ressourcen (1) und dem Einsatz des http-Protokolls (2) wird erreicht, dass jede Anfrage an einen REST-WS alle relevanten Zustandsinformationen enthält. Die Kommunikation wird deshalb als zustandslos bezeichnet.

2.2 Vorgehen zur Entwicklung von RESTful Web-Services

Ressourcen bilden in REST-WS die zentrale Komponente der Systemarchitektur. Ihre Analyse und ihr Entwurf sind grundlegende Voraussetzung für die methodisch geleitete Entwicklung einer REST-Anwendung. Richardson und Ruby schlagen für die Spezifikation von REST-WS folgende Schritte vor (siehe [14]):

1. Identifikation zu speichernder Datenobjekte und anschließende Ableitung von Ressourcen aus diesen Datenobjekten. Ausgangspunkt bilden hierbei Anwendungsfälle und fachliche Anforderungen an das zu entwickelnde System.

Wiederhole für jede Ressource:

2. Festlegung eines eindeutigen Identifikators (URL) und der http-Operatoren (einheitliche Schnittstelle).
3. Entwurf der Repräsentationen und Integration der Ressource und bereits existierender Ressourcen (Verlinkung).
4. Entwicklung und Durchführung von Ablauftests und der Fehlerbehandlungen für das System.

Das beschriebene Vorgehen ist die Basis für eine Vielzahl von Entwicklungsansätzen in der REST-Literatur (vgl. [1], [4], [18], [19]). Ähnlich den objektorientierten Entwurfsmethoden werden zentrale Konzepte in der fachlichen Analyse durch eine Suche nach Substantiven identifiziert und als Ressourcen abgebildet (vgl. [14]). Methodisch bedeutsame Weiterentwicklungen dieses Ansatzes sind nicht bekannt.

3 Ansatz zur modellbasierten Spezifikation einer RESTful SOA

Der konzipierte Ansatz zur Entwicklung einer RESTful SOA besteht aus den Ebenen des Geschäftsprozessmodells (1), des Anwendungsmodells (2) und der REST-Architektur (3) mit jeweils einer struktur- und einer verhaltensorientierten Sicht (vgl. Bild 1). Zwischen den Ebenen bestehen Transformationsbeziehungen (T1 und T2).

Die Modellierung des Geschäftsprozessmodells basiert auf der Notation der SOM-Methodik (vgl. [6]). Für die Ableitung des Anwendungsmodells wird das dort vorgeschlagene Vorgehen verwendet. Eine ausführliche Einführung in Architektur, Vorgehensmodell und Notationen der SOM-Methodik findet sich in [7]. Die für diesen Beitrag wichtigen Grundlagen der Geschäftsprozess- und fachlichen AwS-Modellierung sowie die dazugehörige Notation werden im Rahmen der Fallstudie in Kapitel 4 vorgestellt und erläutert.

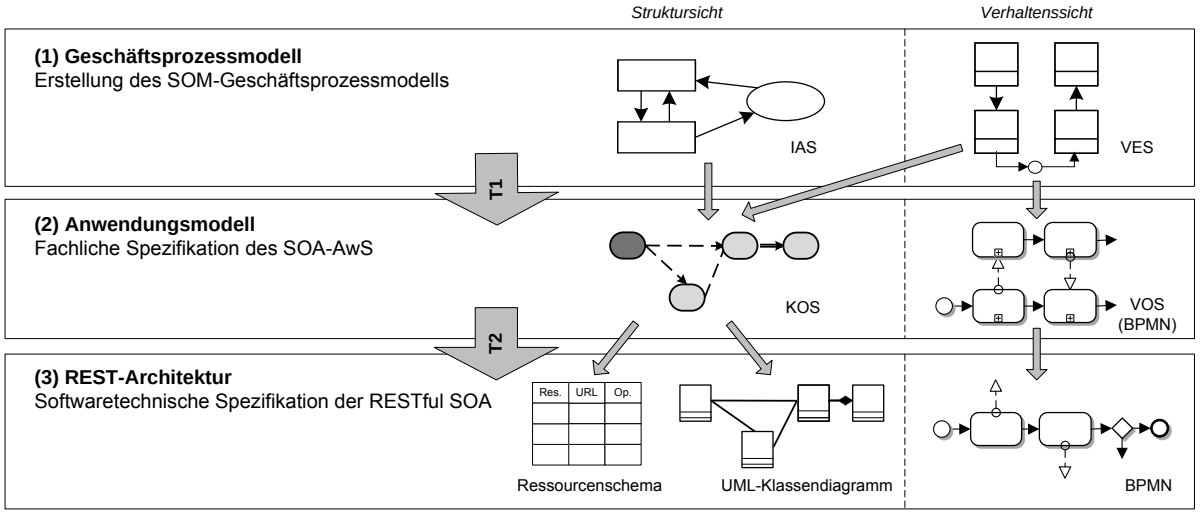

Bild 1: Vorgehen zur modellbasierten Entwicklung einer RESTful SOA

Das gewählte Vorgehen wird zunächst allgemein beschrieben. Eine fallbasierte Anwendung erfolgt in Kapitel 4.

(1) Geschäftsprozessmodell: Auf der ersten Ebene wird das SOM-Geschäftsprozessmodell erstellt und verfeinert. Ausgehend von einem initialen Geschäftsprozess werden das strukturorientierte Interaktionsschema (IAS) und das dazu korrespondierende verhaltensorientierte Vorgangs-Ereignis-Schema (VES) schrittweise entwickelt. Das IAS erfasst die an der Durchführung von Aufgaben beteiligten betrieblichen Objekte und beschreibt deren Koordination mittels Transaktionen. Unter einem betrieblichen Objekt wird eine Menge von Aufgaben verstanden, die mit zugehörigen Sach- und Formalzielen auf einem gemeinsamen Aufgabenobjekt durchgeführt werden. Die Koordination zwischen betrieblichen Objekten kann hierarchisch (Steuerung, Kontrolle) und nicht-hierarchisch (Anbahnung, Vereinbarung, Durchführung) erfolgen. Das Verhalten der betrieblichen Objekte wird im VES in Form eines ereignisgesteuerten Ablaufs der Aufgabendurchführung erfasst (vgl. [6]).

(2) Anwendungsmodell: Im SOM-Geschäftsprozessmodell wird der mit einer SOA zu unterstützende Teil abgegrenzt und in eine fachliche AwS-Spezifikation transformiert (Transformationsschritt T1) (vgl. [7]). Das konzeptuelle Objektschema (KOS) beschreibt die fachliche Struktur des AwS in Form von miteinander in Beziehung stehenden konzeptuellen Objekttypen (OTs). Das Vorgangsobjektschema (VOS) beschreibt das Lösungsverfahren zur Durchführung der Aufgaben und wird als Workflowschema in der Sprache Business Process Model and Notation (BPMN) modelliert (vgl. [13]). Das Lösungsverfahren wird dabei anhand einer Menge von Aktivitäten spezifiziert. Aktivitäten können unter Nutzung der Operatoren der konzeptuellen OTs realisiert werden. Die Transformationsschritte für eine Ableitung des KOS aus dem IAS werden in [7] und die modellbasierte Ableitung des BPMN-Workflowschemas aus dem VES in [13] erläutert.

(3) REST-Architektur: Die softwaretechnische Spezifikation der RESTful SOA erfolgt auf Basis des fachlichen AwS-Entwurfs. Aus den konzeptuellen OTs des fachlichen KOS werden die Ressourcen der REST-WS samt URL und Operatoren abgeleitet und in ein initiales Ressourcenschema überführt. Dieser Transformationsschritt T2 ist durch eine Reihe von Ableitungsregeln beschreibbar (siehe Abschnitt 4.3). Das KOS wird in einem zweiten Schritt in ein UML-Klassendiagramm transformiert. Jeder konzeptuelle OT wird dabei in eine Klasse

überführt, wobei jede Klasse die Struktur einer Ressource anhand von Attributen und Operatoren spezifiziert. Das BPMN-Workflowschema wird übernommen, die Aktivitäten genauer spezifiziert und um Aufrufbeziehungen zwischen Ressourcen als Lösungsverfahren zur Durchführung der Aufgaben erweitert.

Eine Unterstützung der Modellierungstätigkeit des beschriebenen Entwicklungsprozesses kann durch den Einsatz geeigneter Werkzeuge, wie z.B. dem SOM-Modellierungswerkzeug (vgl. [3]), erfolgen. Auf jeder Modellebene findet eine Überarbeitung der erstellten oder initial abgeleiteten Struktur- und Verhaltensmodelle statt. Diese Überarbeitungstätigkeit kann durch geeignete Modellierungswerkzeuge unterstützt werden, ist aber nicht automatisierbar. Insbesondere auf der Ebene des Anwendungsmodells und der REST-Architektur bestehen bezüglich der Spezifikation des fachlichen bzw. softwaretechnischen Lösungsverfahrens für das AwS hohe Freiheitsgrade. Die vorgeschlagenen Ebenen und dort enthaltenen Modelle geben dem Entwickler ein Hilfsmittel an die Hand, um die Komplexität des Entwicklungsprozesses einer RESTful SOA besser beherrschen zu können.

4 Erläuterndes Fallbeispiel

Der Einsatz des Entwicklungsprozesses soll anhand eines Fallbeispiels demonstriert werden. Gegenstand der Fallstudie ist das Unternehmen „Online-Großhandel GmbH“, welches Produkte über eine Onlineplattform an seine Geschäftspartner vertreibt. Der Geschäftsprozess des Unternehmens beschreibt die Distribution von Gütern an seine Kunden. Die Durchführung der Aufgaben des Geschäftsprozesses soll möglichst vollständig durch den Einsatz einer RESTful SOA automatisiert werden.

4.1 Spezifikation des Geschäftsprozessmodells (Ebene 1)

Die Umsetzung des Geschäftsprozesses „Güterdistribution“ erfolgt durch die nichthierarchische Koordination der betrieblichen Objekte Online-Großhandel und Kunde. Der Geschäftsprozess wird in die Phasen Anbahnung (A), Vereinbarung (V) und Durchführung (D) zerlegt. Bild 2 zeigt die initiale Struktur des Geschäftsprozesses und die erste Zerlegungsstufe als IAS sowie das dazu korrespondierende VES.

Nach der Übermittlung der Produktinformationen (A) erfolgt die Bestellung der gewünschten Produkte (V). Die Durchführung der Leistungsübergabe wird mit Lieferung der bestellten Ware vollzogen (D). Zu jedem IAS existiert ein korrespondierendes VES, das den ereignisgesteuerten Ablauf der Aufgaben des Geschäftsprozesses darstellt. Eine Aufgabe wird im VES einem betrieblichen Objekt zugeordnet. Die Modellierung des VES erfolgt in Form eines modifizierten Petri-Netzes (vgl. [13]). Aufgaben werden als Übergänge (Transitions) dargestellt. Die ereignisgesteuerte Ausführung der Aufgaben erfolgt durch das Schalten zulässiger Übergänge. Die beiden Übergange, welche die an einer betrieblichen Transaktion (V: Bestellung) beteiligten Aufgaben (z.B. Bestellung> (senden) und >Bestellung (empfangen)) darstellen, schalten synchron. Deshalb wird zwischen diesen kein Zustand (Place) modelliert (vgl. hierzu Bild 2).

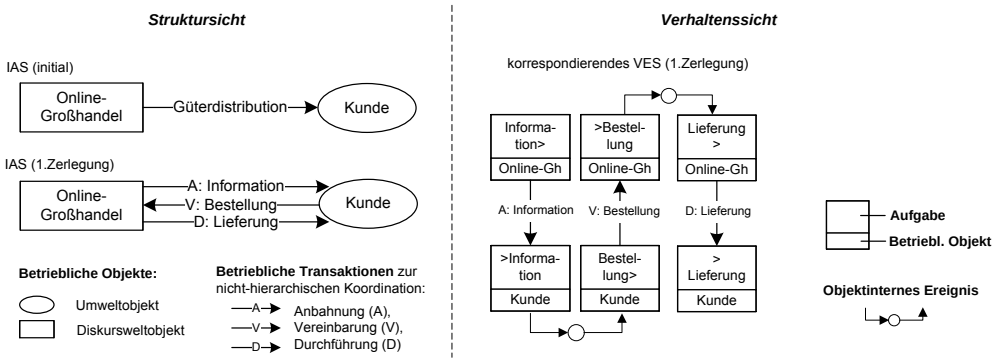

Bild 2: Geschäftsprozess Güterdistribution (initiales IAS und 1. Zerlegung)

Die Interaktion mit dem Kunden wird nun durch Transaktionszerlegung weiter verfeinert (2. Zerlegungsschritt). Die Vereinbarung der Leistungsdurchführung erfolgt in drei sequentiellen Schritten (Produktauswahl, Bestellangebot und Bestellung). Die eigentliche Durchführung der Leistungserstellung besteht aus der Lieferung der Produkte sowie der anschließenden Abrechnung (Rechnung und Zahlung). In einem weiteren Schritt (3. Zerlegung) erfolgt die Identifikation der an der Durchführung des Distributionsprozesses beteiligten betrieblichen Objekte durch Zerlegung des Diskursweltobjekts Online-Großhandel in Vertrieb, Versand, Finanzwesen und Customer Relationship Management (CRM). Der Vertrieb koordiniert die Bestellabwicklung mit dem Kunden und die an der weiteren Durchführung der Bestellung beteiligten Diskursweltobjekte nicht-hierarchisch. Das CRM stellt Kundeninformationen für Finanzwesen und Vertrieb bereit und erbringt somit für diese Objekte eine Serviceleistung. Bild 3 fasst die Ergebnisse der 2. und 3. Zerlegung für IAS und VES zusammen. Eine ausführliche Erläuterung der Regeln zu Objekt- und Transaktionszerlegung findet sich in [7].

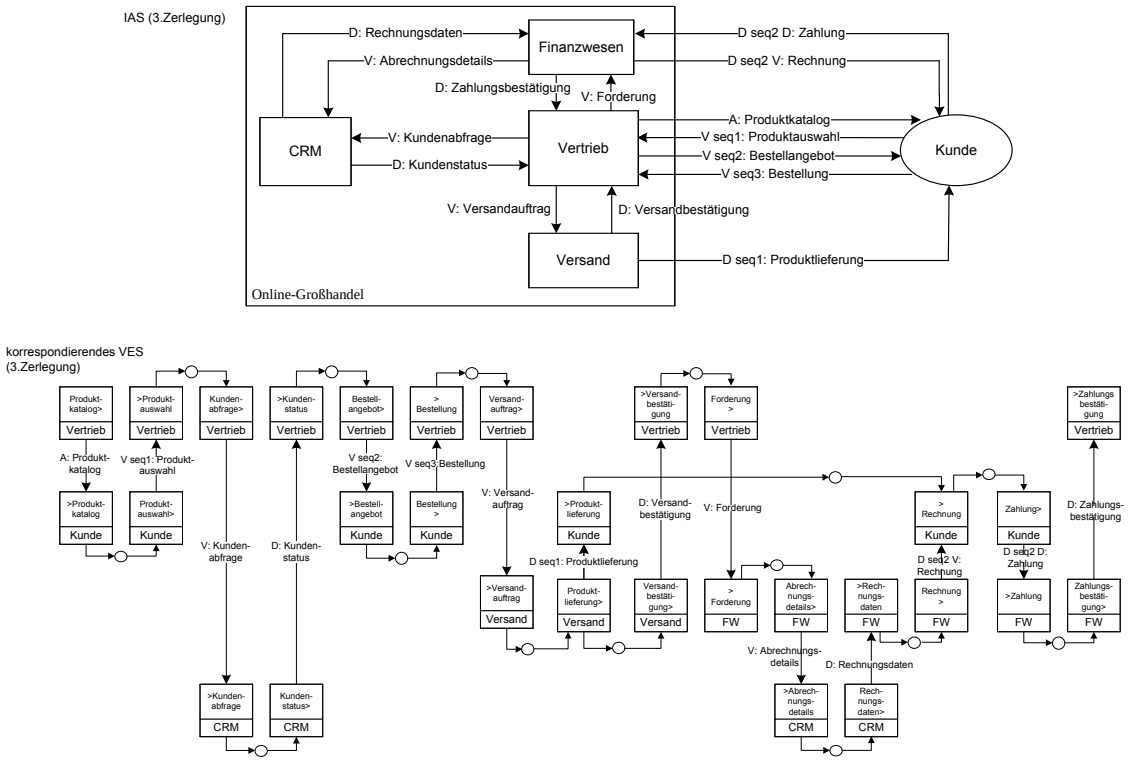

Bild 3: IAS und VES nach Transaktions- und Objektzerlegung (3. Zerlegungsstufe)

4.2 Spezifikation des Anwendungsmodells (Ebene 2)

Aus dem SOM-Geschäftsprozessmodell wird die fachliche Spezifikation des AwS in Form von KOS und VOS abgeleitet (Transformationsschritt T1). Von technischen Aspekten wird hierbei noch abstrahiert. Die Ableitung und Überarbeitung von KOS und VOS erfolgt dabei parallel. In der vorliegenden Arbeit wird aus Gründen der Übersichtlichkeit jedoch mit der Entwicklung des KOS begonnen. Das KOS spezifiziert die fachliche Struktur des AwS anhand einer Menge von miteinander in Beziehung stehenden konzeptuellen OTs.

Die Regeln zur Überführung von IAS in ein initiales KOS werden in [7] beschrieben. Diskurswelt- und Umweltobjekte des IAS werden in objektspezifische OTs und jede Leistungsspezifikation des Geschäftsprozesses in einen leistungsspezifischen OT abgebildet. Objektspezifische und leistungsspezifische OTs sind existenzunabhängig und werden links im KOS angeordnet. Transaktionen im IAS werden in transaktionsspezifische OTs überführt und nach ihrer Abhängigkeit zu den bereits abgeleiteten OTs von links nach rechts im Schema angeordnet. In diese Auswertung werden Ablaufbeziehungen des VES mit einbezogen. Das Ergebnis der initialen Ableitung stellt Bild 4 dar.

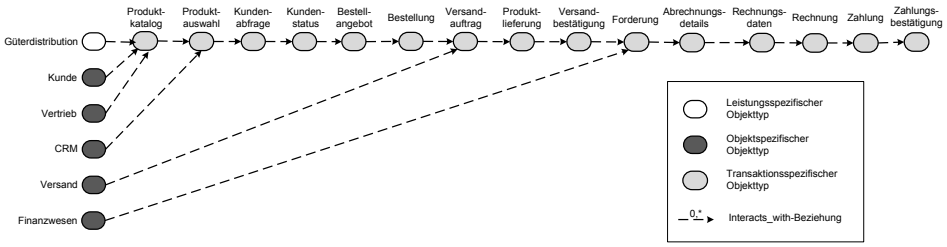

Bild 4: Initiales KOS

Das initiale KOS wird nun überarbeitet (vgl. [7]). Zunächst werden konzeptuelle OTs nichtautomatisierbarer Aufgaben und Transaktionen entfernt. Anschließend werden die Beziehungen zwischen den verbleibenden OTs genauer spezifiziert und diesen OTs Attribute, Nachrichtendefinitionen und Operatoren zugeordnet (vgl. Auftrag in Bild 5). Konzeptuelle OTs, die sich in ihren Attributen oder Operatoren weitgehend überlappen, werden zusammengefasst. Das Ergebnis der Überarbeitung zeigt Bild 5.

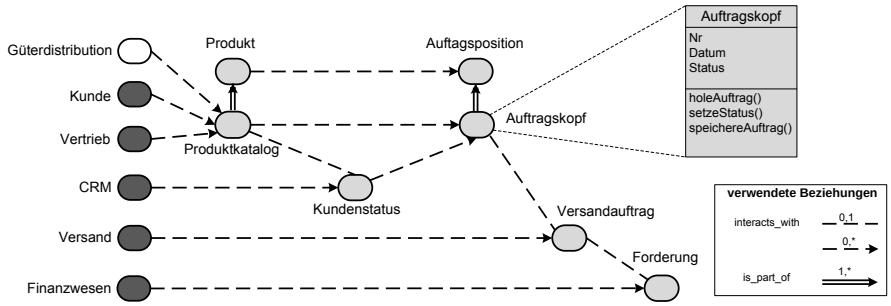

Bild 5: Überarbeitetes KOS

Der Produktkatalog umfasst die angebotenen Produkte. Produktauswahl, Bestellangebot und Bestellung werden zum Objekttyp Auftrag zusammengefasst und zur Vermeidung von Datenredundanz in Auftragskopf sowie -positionen aufgeteilt (is_part_of-Beziehung). Die Objekttypen Kundenabfrage und Abrechnungsdetails bilden den Kundenstatus.

Nach Ableitung und Überarbeitung des KOS wird das VOS in Form eines BPMN-Workflowschemas erstellt. Ausgangspunkt bildet das VES des SOM-Geschäftsprozessmodells (vgl. Bild 3). Das Ergebnis des Ableitungsschrittes zeigt Bild 6. Pütz und Sinz erläutern wie die modellgetriebene Ableitung von BPMN-Workflowschemata aus SOM-Geschäftsprozessmodellen erfolgt (vgl. [13]). Jedes betriebliche Objekt des VES wird in einen Pool transformiert. Die einzelnen Aufgaben eines betrieblichen Objekts werden entweder in eine Aktivität oder einen Subprozess überführt. Transaktionen zwischen betrieblichen Objekten bilden Message Flows, die jeweils eine sendende Aktivität (z.B. Kundenabfrage>) mit einer empfangenden Aktivität (z.B. >Kundenabfrage) verbinden. Objekt-interne Ereignisse im VES werden als Sequence Flows realisiert.

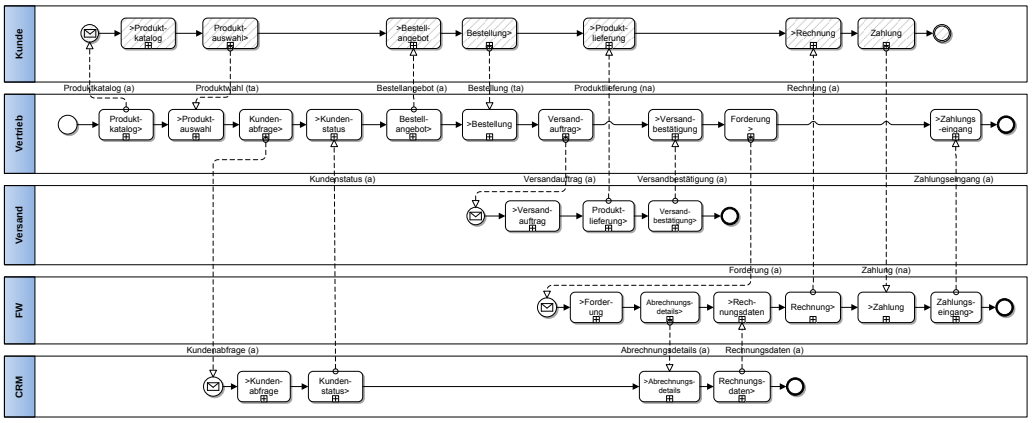

Bild 6: BPMN-Workflowschema

Eine initiale Schnittstellenbeschreibung des AwS erfolgt durch die Spezifikation der Aufrufbeziehungen (Message Flows) zwischen den Pools. Für jeden Message Flow werden hierbei die Nachrichtenflüsse (Transaktionen aus dem VES) sowie der Automatisierungsgrad der Übertragung festgelegt. Es wird zwischen automatisierten (a), teilautomatisierten (ta) und nicht-automatisierten (na) Aufrufen unterschieden (vgl. Bild 6). Die Aktivitäten des Pools Kunde werden nicht durch das Anwendungssystem unterstützt (graue Schraffierung).

4.3 Spezifikation der REST-Architektur (Ebene 3)

Es folgt die softwaretechnische Spezifikation der REST-Architektur auf Basis von KOS und VOS des Anwendungsmodells. Zunächst wird ein initiales Ressourcenschema abgeleitet (siehe in [18] Ressourcentypen). Die Identifikation der Ressourcen, Festlegung deren URL sowie der öffentlichen http-Operatoren erfolgt hierbei auf Basis der überarbeiteten konzeptuellen OTs des fachlichen KOS (vgl. Bild 5). Jeder transaktionsspezifische OT wird in eine (Individual-)Ressource (z.B. *auftraege/{nr}*) sowie dazugehörige Listenressource (z.B. *auftraege/*) überführt. Is_part_of-Beziehungen zwischen zwei transaktionsspezifischen OTs werden als Beziehung Primärressource/Subressource (z.B. *auftraege/{nr}/pos*) realisiert. Objektspezifische OTs können in Einstiegspunkte in die REST-Anwendung überführt werden. Für Listenressourcen werden die Operatoren get und post, und für Individualressourcen get, put und delete festgelegt. Die Ableitung des Ressourcenschemas kann nur initial erfolgen. Es bildet die Basis für die weitere Detaillierung des Schemas durch den Entwickler. Tabelle 1 fasst die Transformationsregeln zusammen.

Überarbeitetes KOS	Ressourcenschema	URL
Objektspez. OT	Einstiegspunkt	Hostname oder Bezeichnung nach Hostname, z.B. *host/* oder *host/einstieg/*
Transaktionsspez. OT	Liste der Ressource	+ Ressourcenname, z.B. *host/ressource/*
Transaktionsspez. OT	Ressource	+ ID zum Namen der Ressourcenliste, z.B. *host/ressource/{id}*
is_part_of-Beziehung	Liste der Subressource	+ Ressourcenbezeichnung zu Ressource, z.B. *host/ressource/{id}/subressource/*
is_part_of-Beziehung	Subressource	+ ID zum Namen der Subressourcenliste z.B. *host/ressource/{id}/subressource/{nr}*
is_a-Beziehung	Liste der Subressource	+ Bezeichnung zur Primärressource
is_a-Beziehung	Subressource	+ ID zur Subressourcenliste

Tabelle 1: Transformationsregeln für die Ableitung des Ressourcenschemas

Als einziger Einstiegspunkt wird für den Nutzer (Kunden) die URL *vertrieb/* bestimmt. Der Produktkatalog fasst eine nutzerabhängige Menge von Produkten zusammen. Die Abfrage der Kundendetails beim CRM erfolgt durch die Angabe eines Query-Parameters (*/?status=*). Die konzeptuellen OTs Kunde und Güterdistribution werden als Nutzer- und Sessionverwaltung des Systems interpretiert. Tabelle 2 zeigt das abgeleitete Ressourcenschema.

Ressource	URL	Operator
Einstiegspunkt	vertrieb/	get
Produktkatalog	vertrieb/produktkatalog/	get, post
Produktliste	vertrieb/produkte/	get, post
Produkt	vertrieb/produkte/{nr}	get, put, delete
Auftragsliste	vertrieb/auftraege/	get, post
Auftrag	vertrieb/auftraege/{nr}	get, put, delete
Auftragspositionsliste	vertrieb/auftraege/{nr}/pos/	get, post
Auftragsposition	vertrieb/auftraege/{nr}/pos/{posnr}	get, put, delete
Kundenstatusliste (CRM)	crm/kunden/	get, post
Statusinfo. eines Kunden	crm/kunden/{nr}/	get, put, delete
Detailinfo. eines Kunden	crm/kunden/{nr}/?status=details	get
Rech.info eines Kunden	crm/kunden/{nr}/?status=rechdaten	get
Versandauftragsliste	versand/versandauftraege/	get, post
Versandauftrag	versand/versandauftraege/{nr}	get, put, delete
Forderungenliste	fw/forderungen/	get, post
Forderung	fw/forderungen/{nr}	get, put, delete
Kundenliste (Verwaltung)	kunden/	get, post
Kunde (Verwaltung)	kunden/{nr}	get, put,delete

Tabelle 2: Initiales Ressourcenschema

Das Ressourcenschema sowie das überarbeitete KOS des Anwendungsmodells sind Ausgangspunkt für die Ableitung des UML-Klassendiagramms der RESTful SOA. Jeder konzeptuelle OT des KOS wird in eine Klasse überführt und eine Typisierung der Attribute sowie Bestimmung der http-Operatoren der Objekttypen vorgenommen. Das VOS (BPMN-Schema) des Anwendungsmodells wird übernommen und in Hinblick auf die Kommunikationsbeziehungen zwischen den Aufgaben sowie der ereignisgesteuerten Durchführung des Lösungsverfahrens überarbeitet. Für die Detaillierung des BPMN-Workflowschemas werden die Spezifikationen aus KOS und Ressourcenschema berücksichtigt. Einen Ausschnitt aus BPMN-Workflowschema und UML-Klassendiagramm zeigt Bild 7.

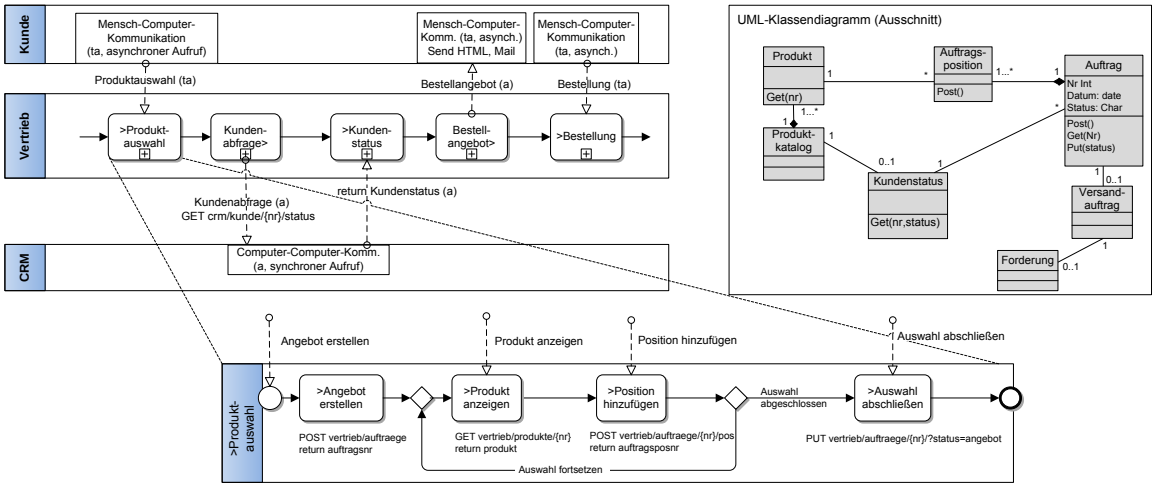

Bild 7: Ausschnitt BPMN-Workflowschema und UML-Klassendiagramm

Eine Festlegung der Repräsentationen einer Ressource ist stark vom konkreten Anwendungsfall abhängig und wird von Fachexperten und Entwicklern getroffen. Hinweise auf den Entwurf der Ressourcenverlinkung geben Aufrufbeziehungen im Workflowschema sowie die Beziehungen zwischen den Objekttypen des KOS. Die Spezifikation von Fehlerbehandlungen und von Testszenarien erfolgt anhand des BPMN-Schemas.

5 Verwandte Arbeiten

Einen Ansatz zur modellbasierten Spezifikation von RESTful Service APIs stellen Laitkorpi et al. in [9] vor. Den Ausgangspunkt bildet dabei ein UML-Sequenzdiagramm, das die funktionale Spezifikation der Problemdomäne darstellt und die fachliche Interaktion zwischen Client und Service erfasst. Ein Metamodell zur Beschreibung von Struktur und Verhalten der Ressourcen in REST-Anwendungen schlägt Schreier vor (vgl. [15]). Xu et al. modellieren Status- und Ablaufinformationen von REST-Anwendungen unter Nutzung von Aktivitätsdiagrammen (vgl. [21]). Die genannten Ansätze betrachten jedoch keine durchgängige Entwicklung von RESTful SOA aus Geschäftsprozessmodellen.

Die Entwicklung von SOA-AwS ausgehend von Geschäftsprozessmodellen beleuchten dagegen mehrere Ansätze zur Spezifikation von SOAP/WSDL-Services. Thomas et al. stellen in [15] ein Architekturmodell vor, mit dem auf Basis von EPK-Modellen fachliche und softwaretechnische Konfigurationen von SOAP/WSDL-Services abgeleitet werden. Eine, dem vorliegenden Beitrag verwandte Arbeit stammt von Bernhard und Jahn (vgl. [2]). Ausgehend von einem SOM-Geschäftsprozessmodell wird in sieben Schritten die Spezifikationen von SOAP/WSDL-Services entwickelt. Dabei wird eine operator-zentrierte Sichtweise auf die Spezifikation von Web-Services eingenommen. Die Identifikation und Ableitung von REST-Ressourcen sowie die Spezifikation von REST-WS werden nicht betrachtet. Einen ausführlichen Überblick über Vorgehensmodelle zur Entwicklung service-orientierter Softwaresysteme geben Thomas et al. [17]. Weitere Literatur, in der die modellgetriebene Entwicklung von RESTful SOA aus Aufgabenspezifikationen in Form von Geschäftsprozessmodellen sowie die modellbasierte Identifikation von REST-Ressourcen der REST-WS untersucht wird, ist nicht bekannt.

6 Zusammenfassung und Ausblick

Der vorliegende Beitrag beschreibt, wie die modellgetriebene Spezifikation von RESTful SOA aus Geschäftsprozessmodellen erfolgen kann. Ergebnis der Arbeit ist ein Prozess, der eine REST-Architektur schrittweise auf drei Abstraktionsebenen entwickelt. Zur Überwindung der semantischen Lücke zwischen der fachlichen Aufgabenträgerebene und den Anforderungen an die softwaretechnische Spezifikation einer REST-Architektur werden Transformationsregeln für die Ableitung der REST-Ressourcen angegeben. Die betrieblichen Transaktionen des Geschäftsprozessmodells bilden dabei die fachliche Basis für die Identifikation der Ressourcen. Ergebnis des Entwicklungsprozesses ist die Spezifikation einer RESTful SOA anhand von Ressourcenschema, UML-Klassendiagramm und BPMN-Workflowschema. Anhand einer Fallstudie erfolgte eine erste Validierung des Ansatzes. Das von Richardson und Ruby beschriebene Vorgehen (Abschnitt 2.2) wird, für die Schritte 1 und 2 vollständig und für die Schritte 3 und 4 teilweise, mit dem konzipierten Ansatz unterstützt.

Die Beschreibung der softwaretechnischen Architektur soll in einer weiteren Arbeit verfeinert werden. Beispielsweise integriert das abgeleitete BPMN-Workflowschema die Innen- und Außensicht der REST-Anwendung. Eine Trennung beider Sichten kann zur Reduktion der aktuell noch hohen Modellkomplexität beitragen. Die Beschreibung der abgeleiteten Ressourcen kann durch Angabe erweiterter Merkmale, wie z.B. alternativer Ressourcentypen oder Repräsentationen, präzisiert werden. Die Überarbeitung der softwaretechnischen Spezifikation wurde in diesem Beitrag beispielhaft für eine Fallstudie durchgeführt. Bei dieser Überarbeitung handelt es sich zwar um eine modellgestützte, aber dennoch manuelle Tätigkeit. Der Systementwickler kann hierbei z.B. durch die Vorgabe von Überarbeitungsregeln besser unterstützt werden. Nicht Teil dieser Arbeit sind die Betrachtung des Übergangs auf die Implementierungsebene sowie die Diskussion von Anforderungen, die sich aus Implementierungsalternativen an eine softwaretechnische Beschreibung der REST-Anwendung ergeben können. Die Spezifikation der RESTful SOA soll in einem weiteren Beitrag als Ausgangspunkt für eine prototypische Implementierung dienen. Somit kann eine Erweiterung und stärkere Validierung des Ansatzes erfolgen. Weitere Fallstudien sind hierzu notwendig. Eine metamodellbasierte Integration von fachlicher und software-technischer Ebene erscheint vielversprechend.

7 Literatur

[1] Allamaraju, S (2010): RESTful Web Services Cookbook. O'Reilly, Sebastopol.

[2] Bernhard, R; Jahn, BU (2009): Modellgetriebene Entwicklung von Serviceorientierten Architekturen. In: Hansen, HR; Karagiannis, D; Fill, HG (Hrsg.), *Wirtschaftinformatik Proceedings 2009*. Wien.

[3] Bork, D; Sinz, EJ (2011): Ein Multi-View-Modellierungswerkzeug für SOM-Geschäftsprozessmodelle auf Basis der Meta-Modellierungsplattform ADOxx. In: Sinz, EJ; Bartmann, D; Bodendorf, F; Ferstl, OK (Hrsg.), *Dienstorientierte IT-Systeme für hochflexible Geschäftsprozesse*: 369-384. University of Bamberg Press, Bamberg.

[4] Burke, B (2009): RESTful Java with JAX-RS. O'Reilly, Sebastopol.

[5] Erl, T (2008): SOA: Principles of Service Design. Prentice Hall, Upper Saddle River.

[6] Ferstl, OK; Sinz, EJ (1995): Der Ansatz des Semantischen Objektmodells (SOM) zur Modellierung von Geschäftsprozessen. Wirtschaftsinformatik 37(3):209-220.

[7] Ferstl, OK; Sinz, EJ (2008): Grundlagen der Wirtschaftsinformatik. 6. Auflage. Oldenbourg, München.

[8] Fielding, R (2000): Architectural Styles and The Design of Network-based Software Architectures. PhD thesis, University of California, Irvine.

[9] Laitkorpi, M; Selonen, P; Systä, T (2009): Towards a Model-Driven Process for Designing ReSTful Web Services. In: *ICWS '09 Proceedings of the 2009 IEEE International Conference on Web Services*: 173-180. Los Angeles.

[10] MacKenzie, CM; Laskey, K; McCabe, F; Brown, PF; Metz, R (2006): Reference Model for Service Oriented Architecture 1.0 – OASIS Standard. http://docs.oasis-open.org/soa-rm/v1.0/soa-rm.pdf. Abgerufen am 01.12.2011.

[11] Melzer, I (2010): Service-orientierte Architekturen mit Web Services: Konzepte – Standards – Praxis. 4. Auflage. Spektrum Akademischer Verlag, Heidelberg.

[12] Pautasso, C; Zimmermann, O; Leymann, F (2008): RESTful Web Services vs. Big Web Services: Making the Right Architectural Decision. In: *Proceedings of the 17th World Wide Web Conference.* Beijing, China.

[13] Pütz, C; Sinz, EJ (2010): Modellgetriebene Ableitung von BPMN-Workflowschemata aus SOM-Geschäftsprozessmodellen. In: Engels G, Karagiannis D, Mayr HC (Hrsg.), *Proceedings Modellierung 2010*. Köllen, Bonn.

[14] Richardson, L; Ruby, S (2007): RESTful Web Services. O'Reilly, Sebastopol.

[15] Schreier, S (2011): Modeling RESTful applications. In: Pautasso C; Wilde, E (Hrsg.), *Proceedings of the Second International Workshop on RESTful Design*. Hyderabad.

[16] Thomas, O; Leyking, K; Dreifus, F; Fellmann, M; Loos, P (2007): Serviceorientierte Architekturen: Gestaltung, Konfiguration und Ausführung von Geschäftsprozessen. Veröffentlichungen des Instituts für Wirtschaftsinformatik, Saarbrücken.

[17] Thomas, O; Leyking, K; Scheid, M (2009): Vorgehensmodelle zur Entwicklung service-orientierter Softwaresysteme. In: Hansen, HR; Karagiannis, D; Fill, HG (Hrsg.), *Wirtschaftinformatik Proceedings 2009*. Wien.

[18] Tilkov, S (2009): REST UND HTTP: Einsatz der Architektur des Web für Integrations-szenarien. Dpunkt.verlag, Heidelberg.

[19] Vinoski, S (2008): RESTful Web Services Development Checklist. IEEE Internet Computing, 12(6):94-96.

[20] Vinoski,S (2008): Serendipitous Reuse. IEEE Internet Computing 12(1):84-87.

[21] Xu, X; Liming, Z; Yan, L; Staples, M (2008): Resource-Oriented Architecture for Business Processes. In: *Proceedings of the 15th Asia-Pacific Software Engineering Conference*: 395-402. Beijing, China.